Auddard & Co.
11/28/2023
MINUTES

Summary of Tasks/Deliverables

Deliverables:

1. Common software engineering artifacts
    a. Project management plan
    b. Requirements document
    c. Design document**
        i. Seamlessly aligns with the specified requirements
    d. Test cases**
2. Well documented C++ program that can evaluate arithmetic expressions with the specified operators and features
3. A user manual or README file explaining how to use the program, including examples

Project Tasks:

1. Expression Parsing:
    a. Implement a function to tokenize the input expression
    b. Create a data structure, such as a stack or a tree, to represent the expression's structure
2. Operator Precedence:
    a. Define the precedence of the operators according to the PEMDAS rules
    b. Implement the logic to evaluate the expression while considering operator precedence
3. Parenthesis Handling
    a. Develop a mechanism to identify and evaluate expressions within parentheses
4. Numeric Constants:
    a. Recognize numeric constants in the input. Initially assume the input will be integers only. In the future, there may be a request to accommodate floating point input values also so you experience change requests and embracing changes
5. User Interface:
    a. Create a user-friendly and legible command-line interface that allows users to enter expressions and displays the calculated results
6. Error handling:
    a. Implement robust error handling to manage scenarios like division by zero or invalid expressions

November 28th, 2023

Meeting Commenced: 3:56 PM
Meeting Adjourned: 4:47 PM
Location: LEEP2 @1320
Attendance:

- ☑ ~~Sriya Annem~~
- ☑ ~~Lauren D'Souza~~
- ☑ ~~Ella Nguyen~~
- ☑ ~~Morgan Nguyen~~
- ☑ ~~Audrey Pan~~
- ☑ ~~Tanu Sakaray~~

Additional Notes:

Meeting Goals/Agenda:

- **Oh man**

Weekly In-Person Meetings for Coding Portion of Project

- ~~Week of the 12th:~~
  - ~~11.14.23 @ 7 PM 4PM~~
    - ~~LEEP2 1320~~
- ~~Week of the 19th:~~
  - ~~11.20.23 @ 5 PM 9AM-10:30AM~~
    - ~~LEEP2 1320 (Confirmed)~~
- Week of the 26th
  - 11.28.23 @ ~~7 PM~~ 4PM
    - LEEP2 1320 (Confirmed)

On Dec 3rd (OPTIONAL)

Replit Link:

- https://replit.com/join/rpgszwquii-ellanguyen3

Test cases Doc

a. FINISH UP
b. Everyone needs to have 5 test cases and check whether they work or not
c. Sriya will write the Purpose section

Implementation:

1. Accuracy: The code should consistently yield accurate results for all valid input scenarios.
   a. Completed: all test cases in the project description work
2. Efficiency: Strive for resource-efficient coding practices, optimizing memory and CPU usage to enhance performance.
   a. Pass by reference instead of pass by value
   b. Destructor?
3. Robustness: The code should gracefully handle invalid input, employing proper error handling techniques to prevent crashes or unexpected behavior.

    a. Adding while loop for when an error occurs so that the user can try again and the program doesn't just end
    b. Error messages with try-catch block
4. Modularity: Design the codebase to be modular, breaking it into well-defined components. This promotes ease of comprehension, maintenance, and scalability.
    a. Multiple functions that split input into tokens and then function that can read it and then function that can interpret it
    b. Do not have multiple classes
    c. Tokenize expressions!!
5. Formatting and Style: Maintain a well-structured and consistent coding style throughout the project, ensuring readability and conformity to coding conventions.
    a. COMPLETE
6. Documentation: Provide clear and concise comments and documentation within the codebase to facilitate understanding and maintainability.
    a. Completed
7. Error-Free Code: Strive for a codebase that is free of errors and defects through thorough code review and testing.
    a. COMPLETED
8. Comprehensive Testing: Rigorously test the code to ensure it meets all requirements and performs as expected.

User Manual:

The user manual for your software product should be an easy-to-understand guide on how to use the software. It should include the following sections:

- **Introduction:** This section should provide a brief overview of the software, including its purpose, features, and how to install and run it. **SRIYA**
- **Getting started:** This section should provide a step-by-step guide on how to use the software to evaluate arithmetic expressions. It should include instructions on how to enter expressions, how to use the various operators and functions, and how to interpret the results. **AUDREY**
- **Advanced features:** This section should describe any advanced features of the software, such as the ability to save and load expressions, or to define custom variables and functions. **LAUREN**
- **Troubleshooting:** This section should provide a list of common problems, if any, that users may encounter, and how to solve them.**ELLA**

In addition to these sections, the user manual may also include the following:

- **Glossary of terms:** This section should define any technical terms that are used in the manual. **Everyone :)**

- **Examples:** This section should provide examples of how to use the software to evaluate different types of arithmetic expressions. **MORGAN**
- **FAQ:** This section should answer frequently asked questions about the software. **TANU**

The user manual should be written in a clear and concise style, and should be easy to understand for users of all skill levels. It should also be well-organized and easy to navigate.

Here are some specific tips for writing a good user manual:

- Use simple language and avoid jargon.
- Be clear and concise.
- Use step-by-step instructions with screenshots or diagrams if necessary.
- Provide examples of how to use the software.
- Anticipate common problems and provide solutions.
- Test the manual with users to get feedback and make sure that it is easy to understand and use.